European Journal of Operational Research 258 (2017) 315–327

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Dynamic scheduling of aircraft landings

Julia A. Bennell[a,*], Mohammad Mesgarpour[b], Chris N. Potts[b]

[a] Southampton Business School, CORMSIS, University of Southampton, Southampton SO17 1BJ, UK
[b] School of Mathematical Sciences, CORMSIS, University of Southampton, Southampton SO17 1BJ, UK

ARTICLE INFO

Article history:
Received 30 April 2015
Accepted 4 August 2016
Available online 10 August 2016

Keywords:
Runway scheduling
Dynamic programing
Local search
Online problem

ABSTRACT

This paper considers the scheduling of aircraft landings on a single runway. There are time window constraints for each aircraft's landing time, and minimum separation times between consecutive landings, where the separation times depend on the weight classes of the two landing aircraft. A multi-objective formulation takes account of runway throughput, earliness and lateness, and the cost of fuel arising from aircraft manoeuvres and additional flight time incurred to achieve the landing schedule. The paper investigates both the static/off-line problem where details of the arriving flights are known in advance, and the dynamic/on-line problem where flight arrival information becomes available over time. Under dynamic scheduling, the algorithm makes periodic updates to the previous schedule to take into account the aircraft that are newly available. We investigate dynamic programing and local search implementations for the static and dynamic problem using random test data and real data from London Heathrow airport.

1. Introduction

According to projections, air transportation demand is expected to grow annually at rates between three and five percent in spite of the recent economic recession. Increasing traffic causes high congestion in the terminal areas, holding delays for arriving aircraft and long queues at the holding departure areas. Given the current congestion levels in the busier airports, accommodating further flights presents a significant challenge. Airport runway capacity is often a limiting factor when creating plans to offer additional flights at an airport. This is because improvements to the management of en-route air traffic have shifted the bottleneck from en-route airspace to the airport (Soomer & Franx, 2008), and more specifically to the runway. Although airport capacity can be increased by building a new runway, making the best use of the existing runway(s) through careful scheduling may reduce the need to improve the infrastructure. In this paper, our focus is on the efficient scheduling of landing aircraft, or specifically the aircraft landing problem (ALP). This is a significant sub-problem of the more general scheduling and routing of aircraft in the terminal manoeuvering area including the resolution of potential aircraft conflicts (Samà, D'Ariano, D'Ariano, & Pacciarelli, 2014).

One of the main factors affecting runway usage is the enforcement of minimum separations between landing aircraft that arise from safety considerations. Wake vortices are rotating masses of air that are generated by aircraft as a consequence of their lift. They can provide a hazard for a following aircraft if it is within a certain distance. Wake vortices are bigger if they are created by a larger aircraft. Moreover, they have a greater impact when the following aircraft is lighter than the leading aircraft. Thus, the required minimum separations between aircraft depend on the weight class of the leading and following aircraft. Consequently, effective scheduling will aim to avoid a light aircraft landing immediately after a heavy aircraft.

In addition to issues of safety, which is the responsibility of air traffic controllers (ATCs), there are other stakeholders with an interest in how aircraft landings are scheduled. Punctuality is a concern for airlines and airports. Airport operations such as gate assignment and baggage handling require careful planning in advance, and delays to an aircraft landing may have a detrimental effect on similar operations for subsequent aircraft. Airlines also prefer schedules that minimize the cost of fuel, and governments typically have targets for reducing CO_2 emissions. Long queues and additional manoeuvres by aircraft to create a landing sequence may increase emissions. ATCs organize the landing of aircraft to meet safety requirements and maximize throughput. Ideally, the aims of all of the various stakeholders would also be taken into account when scheduling the landings of aircraft.

Bennell, Mesgarpour, and Potts (2011) provide an extensive review of airport runway scheduling, which includes numerous studies on the aircraft landing problem. However, most of the research on scheduling aircraft landings deals with the static or off-line

* Corresponding author. Fax: +44 2380593844.
E-mail addresses: J.A.Bennell@soton.ac.uk (J.A. Bennell), C.N.Potts@soton.ac.uk (C.N. Potts).

http://dx.doi.org/10.1016/j.ejor.2016.08.015
0377-2217/

problem in which all aircraft to be scheduled are known at the outset. However, ATCs work in a dynamic or on-line environment where new aircraft enter the controller's airspace over time. In this dynamic problem, decisions about the landing of earlier aircraft have to be made without knowledge of those that may enter the airspace at a later time. Any system that is designed to support the decision making of ATCs should therefore consider the dynamic problem. Further, a solution of the static problem is not of interest from an operational perspective, although it can be used for planning or as a component of an algorithm for the dynamic problem.

Another shortcoming of many studies in the literature is that the models do not address all of the important issues in a practical decision-making environment. For example, the objective functions within these models typically do not address the concerns of all of the stakeholders, and some of the important operational constraints are often missing. Further, the solution approaches often have excessively long run times relative to the almost instantaneous response required in a decision support system that could be of use to ATCs. Finally, many of the algorithms have not been tested using real data.

In view of the above discussion, there is a need for a model that operates in a dynamic environment and considers more of the constraints that arise in practice. Moreover, the model should adopt a multi-objective approach that considers the interests of the different stakeholders. Our aim is to develop a model that meets these requirements, and to design a dynamic/on-line scheduling algorithm that produces solutions sufficiently quickly that it would be of benefit to ATCs.

The remainder of this paper is organized as follows. In Section 2, we provide some background about air traffic control, and review the main contributions from the literature on algorithms to schedule aircraft landings. Section 3 provides a complete description of the aircraft landing problem, including the constraints, objective function, the assumptions made, and a statement of how the static/off-line and the dynamic/on-line problems differ. Our algorithms for the static and dynamic problems are described in Section 4. A computational evaluation of these algorithms is undertaken in Section 5 using real data from London Heathrow airport and randomly generated data sets. Finally, Section 6 contains some concluding remarks.

2. Preliminaries

2.1. Air traffic control

An Air Traffic Management (ATM) system aims to assure the safe and efficient movement of aircraft from the origin to the destination airport. Safety is achieved by ATCs issuing suitable instructions to pilots to ensure that there are sufficient lateral and vertical separations between all aircraft. These instructions may be for the pilot to change direction, ascend or descend, or to change speed.

ATCs work at the airport traffic control tower, terminal airspace control center and en-route control center. The airport control tower is responsible for ground traffic, and take-off and landing within about 5 nautical miles of the airport and 3000 feet above ground level. The terminal airspace control center handles departures and arrivals up to 40 nautical miles and 10,000 feet from the airport. Finally, the en-route control center deals with traffic outside the terminal manoeuvring area.

Each aircraft has an *unconstrained landing time*, which is the time that it would land when there are no other aircraft to impede its progress to the runway. The unconstrained landing time is determined by the arrival planner system after the aircraft enters the range of the relevant radar. A natural landing sequence is based upon ordering the aircraft in non-decreasing order of their unconstrained landing times. We refer to this sequence as first-come-first-served (FCFS). However, the FCFS sequence may create unnecessarily long separations if aircraft of different weight classes fill successive positions in the sequence. Recall that there are minimum separation times to ensure that wake vortices do not create a safety hazard, and these times depend on the weight classes of the leading and the following aircraft. Thus, the FCFS sequence is typically modified by the controllers to reduce the average separation time between aircraft as much as possible.

There are manoeuvres that ATCs can use to change a landing sequence. Vectoring is the most common manoeuvre to shift an aircraft to a later position in the sequence. Rather than flying a direct route, the aircraft changes direction, and then after a given time it rejoins its original route. Aircraft can also be directed to change their speed in order to achieve certain landing times.

Dear (1976) introduces the concept of *constrained position shifting*. Under constrained position shifting, an aircraft cannot move more than ρ positions from its position in the FCFS sequence, where ρ is a given maximum position shift. Constrained position shifting has two purposes. First, it avoids a situation where an aircraft is continually moved towards the end of the landing sequence due to its characteristics (for example, it might be a light aircraft, which are less common and require a larger separation time when following a medium or heavy aircraft). Second, it helps to avoid an ATC's workload from becoming unmanageable. With large deviations from the FCFS sequence, a controller will have to issue frequent instructions to pilots in order to perform the manoeuvres necessary to achieve the desired landing sequence. As an alternative to constrained position shifting, (Bennell et al., 2011) suggest that adopting *maximum time shifting*, where an aircraft's landing time cannot deviate by more than a given amount from its unconstrained landing time, is preferable. Because the density of air traffic varies during the day, applying constraints on time shifts avoids the variability in a flight's delay that can occur when using constraints on position shifts.

The scheduling of aircraft landings is a dynamic/on-line process. A controller can design a landing schedule only for those aircraft that have entered the relevant airspace sector. When more aircraft enter the controller's sector, the landing schedule can be updated. However, those aircraft sufficiently close to the runway should retain their landing times as defined by the current schedule. Aircraft further from the the runway can be rescheduled, although significant changes to the landing schedule are normally restricted to those aircraft that are furthest away from the runway.

2.2. Literature review

In this section, we review the main algorithmic contributions for scheduling aircraft landings. Assume that there are n aircraft whose landings are to be scheduled. A schedule specifies the respective landing times $LT_1, \ldots, LT_n$ of these aircraft.

As shown by Beasley, Krishnamoorthy, Sharaiha, and Abramson (2000), the ALP can be formulated as a mixed-integer program. Moreover, the close relationship with machine scheduling problems has been noted and exploited by Brentnall (2006). Artiouchine, Baptiste, and Dürr (2008) also the ALP model as a single machine scheduling problem and study the complexity of the problem. We can regard the problem as one of sequencing the aircraft, because the actual landing times are easily computed once the sequence is known.

A variety of authors employ dynamic programing as a solution approach. Many of these studies assume that aircraft within a weight class can be sequenced, typically by establishing that there exists an optimal solution to the ALP that always satisfies this property. The ALP then reduces to one of merging the

individual sequences constructed for the different weight classes, with dynamic programing providing an effective approach for finding an optimal merging. The number of weight classes C is assumed to be fixed (in most practical applications, $C = 3$ or $C = 5$).

The first application of dynamic programing to the ALP is due to Psaraftis (1978, 1980). In this study, all aircraft within the same weight class are identical. Using state variables to indicate the number of aircraft of each weight class that are scheduled and a state variable to indicate the class of the last aircraft in the partial sequence, he derives an algorithms to minimize $\mathrm{LT}_{\max}$, where $\mathrm{LT}_{\max} = \max_{j=1,\ldots,n} \mathrm{LT}_j$, and to minimize $\sum_{j=1}^{n} \mathrm{LT}_j$ with a time complexity of $O(n^C)$. Constraints on position shifting can be incorporated into the algorithm without increasing the time complexity. Also, a further generalization to the case of more than one runway is possible.

Several dynamic programing algorithms for the ALP are presented by Brentnall (2006). For some objective functions, he derives the optimal order of aircraft within each weight class, and is therefore able to generalize the dynamic programing algorithms of Psaraftis. He also develops dynamic programing algorithms for the problem of scheduling landings of aircraft located in several holding stacks, of the type used at London Heathrow airport. His assumption is that the next aircraft to leave any stack must be the lowest one.

Balakrishnan and Chandran (2010) present a dynamic programing framework for runway scheduling that is applicable to the ALP. The algorithm, which minimizes the landing time of the last aircraft, is also shown to generalize to delay-based objective functions. In addition to standard separation constraints, there are constraints on position shifting, precedence constraints between aircraft, and constraints imposed by time windows. Lee and Balakrishnan (2008) extend the previous framework proposed by Balakrishnan and Chandran (2010) and Chandran and Balakrishnan (2007) by presenting a dynamic programing algorithm for minimizing the total delay costs associated with an arrival schedule. Also, they study the problem of minimizing the fuel cost for an arrival schedule, since fuel is a significant factor impacting the profitability of airlines. They allow the earliest landing time to be less than the unconstrained landing time, which is referred to as time advance, and show that up to 3 minutes of time advance is beneficial in most practical cases.

A specialized simplex method is proposed in Ernst, Krishnamoorthy, and Storer (1999) to find the optimal landing schedule, given a partial ordering of the aircraft. They develop a heuristic-based problem space search which consists of the simplex algorithm to compute the optimal landing time, a constructive based heuristic to generate a good sequence, and a genetic algorithm to search the perturbation space. The heuristic algorithm and simplex method are used to obtain upper and lower bounds for a branch-and-bound algorithm which minimizes the sum of delays.

In addition to providing an extensive literature overview, Beasley et al. (2000) present linear programing (LP)-based tree search approaches for scheduling landings on both single and multiple runway problems. The model is based on an earlier mixed-integer programing formulation of Abela, Abramson, Krishnamoorthy, and Silva (1993). Their objective is to minimize the deviation from the unconstrained landing times. Some additional constraints are proposed in order to reduce the search space associated with the mixed-integer formulation and to strengthen the LP relaxation. The static problem is solved optimally for problem instances involving up to 50 aircraft and four runways. Faye (2015) use the Beasley et al. (2000) formulation and present a dynamic constraint generation algorithm. The approach is based on an approximation of the separation time matrix to a rank two matrix, which provides a very good LP relaxation.

Beasley, Sonander, and Havelock (2001) develop a population heuristic addressing the static single-runway ALP with time window restrictions. The algorithm aims to minimize the sum of squares of deviations from unconstrained landing times. Computational results are presented for a single problem instance obtained from observations during a busy period at London Heathrow airport, where minimum separations are based on five weight classes.

Pinol and Beasley (2006) implement two different population heuristics, scatter search and a bionomic algorithm, for the multiple-runway ALP. Two alternative objective functions are proposed based on earliness and lateness with respect to the unconstrained landing time. Specifically, minimization of a linear function comprising weighted earliness plus weighted tardiness, and maximization of a non-linear function comprising the total squared earliness minus the total squared lateness are considered.

A comparative study of algorithms is provided by Fahle et al. (2003). Specifically, they compare a mixed-integer programing formulation with binary variables defining the landing order between each pair of aircraft, an integer program employing time-indexed variables, constraint programing, and local search approaches based on descent and simulated annealing. A satisfiability formulation was also explored, but this only produces a feasible solution rather than searching for an optimal solution. Computational results show that the constraint programing, the mixed-integer and integer programing approaches are computationally expensive on some instances. However, the descent and simulated annealing approaches provide good-quality solutions within reasonable computation times. Soomer and Franx (2008) include airline preferences into the model proposed by Beasley et al. (2000). They evaluate the performance of a mixed-integer programing model and a descent algorithm in which the search is performed over sequences with linear programing used to determine actual landing times from the given sequence.

The dynamic aircraft landing problem in which the aircraft to be scheduled become available to controllers over time has received relatively little coverage in the literature. However, those studies that are available typically use the same general rolling horizon (also referred to as receding horizon) approach. They first select an update time, which is typically a few minutes, when a new schedule is to be created. Any aircraft that have landed when the schedule is updated are removed from the system, any new aircraft that have appeared within the relevant scheduling horizon are added to the system, and then the resulting problem is solved to give a new schedule. Some studies also assume a freeze time when computing an updated schedule. Specifically, any aircraft that is scheduled to land within the freeze time cannot be rescheduled during an update.

Ciesielski and Scerri (1997) investigate the use of genetic algorithms for the dynamic problem of scheduling landings on two runways at Sydney airport. Based on a three-minute update time, positive conclusions are reached about the ability of genetic algorithms to produce schedules of good quality in real time. Beasley, Krishnamoorthy, Sharaiha, and Abramson (2004) introduce a displacement term into their objective function, which penalizes any unfavorable changes when the schedule is updated, where an unfavorable change is one that moves the landing time further away from its preferred landing time. They create their schedules using either the tree search approach of Beasley et al. (2000), one of the heuristics employed within the tree search approach, or the population heuristic of Beasley et al. (2001), where the updated schedule is computed when a new aircraft enters the system. Moser and Hendtlass (2007) propose an extremal optimization approach (essentially a local search approach that aims to improve the worst components of a solution through local modifications) for this problem, although their model does not include the displacement function. Hu and Chen (2005a, 2005b)

develop a framework for rolling/receding horizon approaches. Their computational work indicates that relatively short horizons of between 10 and 20 minutes are sufficient to provide schedules with suitably low delays. Murça and Müller (2015) develop a MILP model for the dynamic sequencing and scheduling of aircraft landings. They consider alternative approach routes to the runway as a means of shortening or lengthening the path, and also consider departing aircraft in their model. They set a scheduling window of 60 minutes and a freeze horizon of 20 minutes.

More recently, there has been a number of papers focusing on traffic management across the entire terminal manoeuvering area (TMA), including arriving and departing aircraft. D'Ariano, Pistelli, and Pacciarelli (2012) state that ATC decisions can be divided into routing decisions, involving the route the aircraft takes through the TMA including holding circles, air segments and runways, and scheduling decisions. They model the problem as a job shop scheduling problem and use an alternative graph formulation. The objective is to minimize delay caused by the resolution of conflicts in the TMA. The scheduling problem is solved using branch and bound, and the routing problem is solved using tabu search. Results show that combining routing and scheduling is beneficial to reducing delay. D'Ariano, Pacciarelli, Pistelli, and Pranzo (2015) build on this work by introducing new practical constraints and testing two greedy heuristics. Samà, D'Ariano, and Pacciarelli (2013) also use the alternative graph approach to solve the scheduling problem assuming fixed routing. They model the dynamic case using a rolling horizon approach. The branch and bound solver could find optimal solutions in most cases within 60 seconds, while FCFS was significantly faster but produced much higher delays. A further paper by the same authors (Samà et al., 2014) introduces a MILP formulation of the scheduling and rerouting problem and provides solution of the model using a commercial solver. Also, two decomposition frameworks are proposed: time-based through a rolling horizon and problem-based by separating the routing and scheduling. Computational results shows that the competitiveness of each approach depends on the instance. Finally, Samà, D'Ariano, D'Ariano, and Pacciarelli, 2016 use the alternative graph model to consider further performance indicators.

In summary, there are numerous studies on the static ALP with dynamic programing, local search and genetic algorithms providing competitive solution approaches. The limited literature on the dynamic ALP is too fragmented for any common themes to emerge.

3. The aircraft landing problem

In this paper, we consider the ALP with a single runway that is used solely for landings. This situation is common, although there are airports where both take-offs and landings are scheduled on the same runway. Associated with any schedule are landing times. Specifically, in any schedule, let LT_j denote the landing time of aircraft j, for $j = 1, \ldots, n$, where n is the number of aircraft in the schedule that is to be created.

In the static/off-line version of the problem, n and all data associated with these aircraft are known in advance of creating the schedule. However, in the dynamic/on-line version of the problem, aircraft arrive into an ATC's airspace over time. In practice, controllers typically have knowledge of an aircraft between 30 and 40 minutes before it can reach the runway. The number of aircraft is not known in advance. Further, no information is available to controllers about aircraft that have yet to arrive into their airspace. Thus, scheduling decisions have to be taken on the basis of partial data.

The model formulation that follows attempts to include an element of the type of coordinated planning to be used in the future by considering the interests of the various stakeholders. Currently, however, ATCs usually schedule landings to minimize separation times between aircraft, subject to meeting safety requirements. In spite of our model's broader remit, a suitable choice of parameters maintains compatibility with the criteria upon which ATCs make their decisions in a current-day setting.

3.1. Constraints

The constraints on the aircraft landing problem are divided into two main types. There are constraints on the time that an aircraft can land, and constraints on the separation time between landings.

3.1.1. Landing time constraints

There are various constraints on the landing time of each aircraft j, for $j = 1, \ldots, n$, that take the form of time windows. First, LT_j should lie within a time window $[elt_j, llt_j]$, where elt_j and llt_j are the earliest and latest landing times of aircraft j. Typically, the earliest landing time is the time aircraft j takes to fly from its current location to the runway at a maximum safe speed. The latest landing time is usually the maximum possible flight time based on the fuel carried by the aircraft, although there could be reasons why an airport or airline could stipulate a smaller value of the latest landing time. Second, LT_j should lie within a time window based on the unconstrained landing time, ult_j, of aircraft j. The value of ult_j is the time that aircraft j would be expected to land when there are no other aircraft to impede its progress to the runway. It is determined by the arrival planner system after the aircraft enters the range of the relevant radar. Aircraft j is assumed not to land before ult_j, but may land up to a maximum time shift ts_j after ult_j, which means that LT_j should lie within the time window $[ult_j, ult_j + ts_j]$.

The two time windows defined by the earliest/latest landing times and the deviations from the ult_j can be combined. This provides a constraint of the form

$$e_j \le LT_j \le l_j \quad \text{for} \quad j = 1, \ldots, n, \tag{1}$$

where $e_j = \max\{elt_j, ult_j\}$ and $l_j = \min\{llt_j, ult_j + ts_j\}$.

An aircraft j may also have an associated preferred landing time plt_j. The preferred landing time may be based on the aircraft's flight plan, the airlines timetable, or a time used by the airport in their plans for assigning a gate to the aircraft or for the baggage to be unloaded. However, we view the preferred landing time as a soft constraint that we address when considering the objective function.

3.1.2. Separation time constraints

Associated with each aircraft is a weight class that determines the minimum separation times between successive landings. Let C denote the number of classes. Also, let s_{bc} be the minimum separation time when an aircraft of class b lands before an aircraft of type c, for b, $c = 1, \ldots, C$. We assume that the separation times satisfy the triangle inequality so that for any aircraft of types a, b and c we have $s_{ab} + s_{bc} \ge s_{ac}$. This implies that it is sufficient to impose the separation time constraints only between successive pairs of aircraft in the landing sequence.

Due to the importance of the separation time constraints, it is sometimes convenient to use double indices for the aircraft. For any weight class c, let n_c denote the number of aircraft in this class, where $n = \sum_{c=1}^{C} n_c$. We then refer to the aircraft in each weight class c as $(1, c), \ldots, (n_c, c)$. Because an aircraft (i, b) lands either before or after any other aircraft (j, c), we obtain a separation constraint

$$LT_{i,b} + s_{bc} \le LT_{j,c} \quad \text{or} \quad LT_{j,c} + s_{cb} \le LT_{i,b} \tag{2}$$

for each pair of aircraft (i, b) and (j, c).

Note that there may be precedence constraints specifying that one aircraft must be placed before another in the landing sequence. Thus, if aircraft (i, b) must land before aircraft (j, c) according to the precedence constraints, then constraint (2) is replaced by $\mathrm{LT}_{i,b} + s_{bc} \leq \mathrm{LT}_{j,c}$.

3.2. Objective function

As previously discussed, the ALP involves a number of stakeholders with various priorities. Therefore, adopting a multi-objective approach is appropriate. Since the problem is complex and our aim is to find solutions quickly enough to solve the online problem, we adopt the approach of forming a weighted sum of the individual objectives.

The main objective of ATCs after taking into account safety is to maximize runway throughput. This naturally translates into minimizing the landing time of the last aircraft in the schedule, or more formally the objective is to minimize $\mathrm{LT}_{\max}$, where $\mathrm{LT}_{\max} = \max_{j=1,\ldots,n} \mathrm{LT}_j$. However, in a more realistic dynamic scheduling environment, there is a high likelihood that the latter part of the schedule will change due to new aircraft arriving, with the result that only the initial part of the landing schedule is implemented. Therefore, focusing only or mainly on the landing time of the last aircraft may create schedules that are less suitable when used for scheduling within a dynamic environment. Thus, we also consider the minimization of the average landing time

$$\mathrm{ALT} = \sum_{j=1}^{n} \mathrm{LT}_j / n, \tag{3}$$

which aims to reduce each of the landing times rather than just the last. Smaller landing times of aircraft early in the sequence may be especially advantageous in a dynamic environment where only the first part of the schedule is executed and the remainder of the schedule is updated as new aircraft join the system. The overall contribution to the objective function of our runway throughput measure is

$$w_1 \mathrm{LT}_{\max} + w_2 \mathrm{ALT}, \tag{4}$$

where w_1 and w_2 are suitably chosen non-negative weights for the maximum and average landing time, respectively.

The notion of a preferred landing time is introduced in Section 3.1.1. For each aircraft j, we define a time window $[\mathrm{plt}_j - \delta_j^e, \mathrm{plt}_j + \delta_j^l]$ within which the aircraft should ideally land, were δ_j^e and δ_j^l define allowable tolerances for earliness and lateness with respect to plt_j, respectively. If $\mathrm{LT}_j < \mathrm{plt}_j - \delta_j^e$, then there is an earliness penalty $u_j^e(\mathrm{plt}_j - \delta_j^e - \mathrm{LT}_j)$, where u_j^e is a penalty per unit of earliness with respect to the left-hand end of the time window. Similarly, if $\mathrm{LT}_j > \mathrm{plt}_j + \delta_j^l$, then there is a lateness penalty $u_j^l(\mathrm{LT}_j - \mathrm{plt}_j - \delta_j^l)$, where u_j^l is a penalty per unit of lateness with respect to the right-hand end of the time window. Generally, we would expect the model parameters to be chosen so that $u_j^l \geq u_j^e$ because lateness usually causes greater disruption then earliness. Thus, the overall penalty for violation of the time windows defined for preferred landing times is

$$\mathrm{TW} = \sum_{j=1}^{n} u_j^e \max\{\mathrm{plt}_j - \delta_j^e - \mathrm{LT}_j, 0\} + \sum_{j=1}^{n} u_j^l \max\{\mathrm{LT}_j - \mathrm{plt}_j - \delta_j^l, 0\}. \tag{5}$$

Lastly, the cost of using more fuel than is necessary for a flight is a concern for airlines, and moreover a reduction in fuel burn is helpful in achieving government targets on CO_2 emissions. Thus, another objective is the minimization of the additional fuel used to achieve a landing schedule. As a baseline, a landing time of ult_j is assumed for each aircraft j. Any later landing for aircraft j, as defined by $\mathrm{LT}_j > \mathrm{ult}_j$, causes the aircraft to use more fuel due to being airborne for longer and also possibly through some manoeuvres requested by the ATC to delay its landing time. Recall that we do not allow any aircraft j to land before ult_j. If v_j^l denotes the cost per unit time of the extra fuel associated with lateness relative to ult_j, then the overall extra fuel cost is

$$\mathrm{EF} = \sum_{j=1}^{n} v_j^l \max\{\mathrm{LT}_j - \mathrm{ult}_j, 0\}. \tag{6}$$

Since the ALP may involve the simultaneous optimization of various dependent objectives that are not necessarily aligned, a trade-off among the objectives is required. Therefore, they need to be optimized in the form of a weighted multi-criteria objective function. Using suitable weights, we can combine the different objectives defined in (4)–(6) to give the overall objective function

$$w_1 \mathrm{LT}_{\max} + w_2 \mathrm{ALT} + w_3 \mathrm{TW} + w_4 \mathrm{EF}, \tag{7}$$

for appropriately chosen non-negative weights w_3 and w_4 (as well as w_1 and w_2). This expression is to be minimized, subject to constraints (1) and (2).

Based on Eq. (7), the incremental cost of aircraft j landing at time t is given by

$$g_{j,t} = w_2 t/n + w_3(u_j^e \max\{\mathrm{plt}_j - \delta_j^e - t, 0\} + u_j^l \max\{t - \mathrm{plt}_j - \delta_j^l, 0\}) + w_4(v_j^l \max\{t - \mathrm{ult}_j, 0\}). \tag{8}$$

3.3. Assumptions

The decision variables in our model are the landing time variables LT_j for $j = 1, \ldots, n$. We assume that any selection of landing times that are chosen to satisfy (1) and (2) define a feasible solution.

One aspect of feasibility that we do not consider is runway occupancy by a landing aircraft. Suppose that the aircraft landing immediately before (j, c) is (i, b). According to constraint (2), aircraft (j, c) could land as early as $\mathrm{LT}_{i,b} + s_{bc}$. Our model assumes that aircraft (i, b) has left the runway by this time. Thus, we do not model the blocking of the runway by any aircraft that has already landed or by any aircraft that is taxiing.

Another operational issue that does not appear in our model concerns the manoeuvres required by aircraft to achieve those landing times that correspond to the values of the decision variables. We aim to avoid the need for excessive resequencing of aircraft by imposing constraint set (1) which incorporates a maximum time shift of each aircraft. On this basis, our assumption is that ATCs can achieve the desired landing times by using relevant techniques (as pointed out by Bennell et al. (2011), vectoring, detour and shortcut are used by ATCs to position aircraft according to the desired landing sequence) and that these manoeuvres can be achieved in the terminal airspace.

3.4. The dynamic problem

The dynamic problem is most critically concerned with the aircraft that will land next, or more specifically, the aircraft that will enter the freeze horizon. However, these decisions are impacted by other aircraft in the schedule; hence, it is appropriate to adopt a rolling horizon approach. Specifically, the static problem is first solved for aircraft within a certain time window that contains the freeze horizon. Then, after a given update period, the static problem is solved again for the aircraft in the new time window. We

create successive time windows so that they overlap, which allows for the possible changes of landing positions between aircraft. Since the information about available aircraft is continuously updated and decision are made in real time, the scheduling algorithms must run within a few seconds.

The above formulation holds for the static problem with n chosen as the total number of aircraft, and for the dynamic problem with n chosen as the subset of aircraft available to the ATC for scheduling at a particular time. In the dynamic aircraft landing problem, aircraft are scheduled for landing using a rolling horizon approach. This means that every τ units of time, for some suitable chosen time interval of length τ, the previously created (provisional) schedule is updated by removing aircraft at the beginning of the schedule that land and therefore leave the system, and by including any new aircraft entering the system that appears on the ATC's radar screen. Some aircraft that are sufficiently close to the start of the schedule are not eligible to be rescheduled for safety reasons. Further, the likelihood of an aircraft being rescheduled reduces as it gets closer to landing. This is because any new aircraft entering the system are too far away to have a significant influence on the selection of landing times of aircraft close to the runway.

We refer to τ as the *update time*. Typically, τ may be approximately five minutes. Too small a value of τ would result in too frequent updates to the schedule, possibly with only one or two additional aircraft in the system. On the other hand, if τ is too large, some of the opportunities for manoeuvres to create better landing schedules may be lost. We investigate different values of τ in our computational experiments.

4. Algorithms for creating landing schedules

This section describes the development of our solution algorithms for the aircraft landing problem. Our goal is to design algorithms that run in under five seconds and preferably provide solutions in under one second. In the following four subsections, we present various search algorithms for solving the static problem. Although being of some independent interest for the static ALP, these algorithms provide the core search mechanism for tackling the dynamic problem. We then describe the solution procedure for the dynamic problem in the final section.

By its nature, decision making for the dynamic problem can never be perfect because some information is unavailable when the various decisions are to be made. On this basis, algorithms for the static problem that are designed for use in solving the dynamic problem do not necessarily have to guarantee optimality. Instead, heuristics that provide good-quality solutions for the static problem at modest computational expense can be used.

4.1. FCFS

In FCFS, the aircraft are sequenced in non-decreasing order of their unconstrained landing times. Thus, the landing sequence σ is chosen so that $\text{ult}_{\sigma(1)} \leq \cdots \leq \text{ult}_{\sigma(n)}$. The landing sequence effectively defines precedences between aircraft that land in succession. Thus, the actual (smallest) landing times are determined by applying constraints (1) and (2) in a straightforward way.

4.2. Dynamic programing

Our dynamic programing algorithm assumes that, within each weight class, the aircraft are ordered in non-decreasing order of their unconstrained landing times. We index the aircraft accordingly, so that $\text{ult}_{1,c} \leq \cdots \leq \text{ult}_{n_c,c}$, for $c = 1, \ldots, C$. We also assume that, for the selected landing sequence, each aircraft is scheduled to land as early as possible subject to the separation constraints (2).

Our proposed dynamic program has stages that are indexed by k, where k denotes the number of aircraft that have landed in the partial schedule. The aircraft within the different weight classes can be viewed as C queues operating in parallel that have to be merged. This merging is achieved by successively selecting the aircraft at the front of one of these queues to be the next to land.

The dynamic program has state variables $(m_1, \ldots, m_C, c, t)$. The first set of state variables are indices $m_1, \ldots, m_C$, where $0 \leq m_b \leq n_b$, indicating that, within the partial schedule, aircraft $(1, b), \ldots, (m_b, b)$ for $b = 1, \ldots, C$ have landed. Thus, the total number of aircraft to have landed is $k = \sum_{b=1}^{C} m_b$. State variable c indicates that aircraft (m_c, c) is the last to land in the partial schedule, and t is this aircraft's landing time. These state variables are sufficient to create and evaluate any continuation of a partial schedule, since the identity of those aircraft to have landed is known and the landing time can be computed for the aircraft that is selected to land next. Specifically, if aircraft $(m_b + 1, b)$ is selected to land next, then its landing time is $\max\{e_{(m_b+1,b)}, t + s_{cb}\}$.

Let $f(m_1, \ldots, m_C, c, t)$ denote the minimum total cost among partial landing schedules corresponding to state $(m_1, \ldots, m_C, c, t)$. Our dynamic programing algorithm can be stated formally as follows.

Algorithm DP

Initialization

Set $k = 1$, and

$$f(1, \ldots, 0, 1, e_{(1,1)}) = g_{(1,1),e_{(1,1)}}$$
$$\vdots$$
$$f(0, \ldots, 1, C, e_{(1,C)}) = g_{(1,C),e_{(1,C)}}$$

where $e_{(1,1)}, \ldots, e_{(1,C)}$ are lower bounds on landing times as used in constraints (1), and the function g is defined in Eq. (8).

Next Stage Generation

For each state $(m_1, \ldots, m_C, c, t)$ such that $\sum_{b=1}^{C} m_b = k$ and each b such that $m_b < n_b$, generate the potential new state $(m_1, \ldots, m_{b-1}, m_b + 1, m_{b+1}, \ldots, m_C, b, t')$, where $t' = \max\{e_{(m_b+1,b)}, t + s_{cb}\}$. If $t' > l_{(m_b+1,b)}$, then the potential new state is infeasible and discarded; otherwise, it is retained and its associated value is $f(m_1, \ldots, m_C, c, t) + g_{(m_b+1,b),t'}$, where $g_{(m_b+1,b),t'}$ is computed from (8).

Next Stage Elimination

If any state $(m'_1, \ldots, m'_C, b, t')$, where $\sum_{c=1}^{C} m'_c = k + 1$, is created more than once in the Next Stage Generation step, select the one with the smallest associated value V and set $f(m'_1, \ldots, m'_C, b, t') = V$. If $k < \sum_{b=1}^{C} n_b$, then set $k = k + 1$ and return to the Next Stage Generation step.

Select Solution

Among all states $(n_1, \ldots, n_C, b, t')$ for $b = 1, \ldots, C$ and all t', select the one with the smallest value of $w_1 t' + f(n_1, \ldots, n_C, b, t')$.

In Algorithm DP, the Initialization considers partial schedules containing one aircraft that may belong to any of the C weight classes. The next Next Stage Generation step appends one aircraft to the current partial landing schedule and evaluates the cost of the new partial schedule if it is feasible. Feasibility of the new partial schedule with respect to the earliest are latest landing time constraints (1) is ensured by selecting $t' \geq e_{(m_b+1,b)}$ and only retaining a partial schedule when $t' \leq l_{(m_b+1,b)}$. In the Next Stage Elimination step, any duplicate states are eliminated on the basis of their cost. Finally, the Select Solution step searches all states corresponding to schedules in which all aircraft have landed, and selects one with minimum cost.

To justify that Algorithm DP produces an optimal landing schedule under the assumptions made, we first observe that all

potential states are generated through the Initialization and Next Stage Elimination steps. Further, the Next Stage Elimination step removes any dominated partial landing schedules through the principle of optimality. Finally, the minimum cost solution selected in Select Solution step ensures that an optimal landing schedule is obtained.

Algorithm DP has n stages and $O(Cn^CT)$ states within each stage, where T denotes the number of potential landing times of the aircraft. However, $T \leq n(n+1)^{C^2}$ because each potential landing time is equal to $e_j + \sum_{b=1}^{C}\sum_{c=1}^{C} x_{bc}s_{bc}$ for some aircraft j, and some $x_{bc} \in \{0, 1, \ldots, n\}$ for $b, c = 1, \ldots, C$. Therefore, the time complexity of Algorithm DP is $O(C^2 n^{C^2+C+2})$, which is polynomial for fixed C.

4.3. Iterated descent

We first describe a descent algorithm that provides the basic building block for our iterated descent method. Solutions are represented as a landing sequence of aircraft. Thus, each solution is defined by some aircraft sequence $\sigma = (\sigma(1), \ldots, \sigma(n))$. We use a combined *insert*, *swap* and *2-insert* neighborhood. Insert and swap are widely-used neighborhoods for a variety of sequencing problems and have been used for the ALP by Fahle et al. (2003), whereas the 2-insert neighborhood has not been used previously. A 3-insert neighborhood was also investigated but resulted in very few feasible moves, and is therefore not included.

The insert neighborhood comprises all sequences that can be obtained from the current sequence by removing an aircraft from its current position and inserting it into a new position in the sequence. Thus, for $1 \leq h < i < j \leq n$, two insert neighbors of σ are

$$(\sigma(1), \ldots, \sigma(h), \sigma(i), \sigma(h+1), \ldots, \sigma(i-1), \sigma(i+1), \ldots, \sigma(n)) \quad (9)$$

$$(\sigma(1), \ldots, \sigma(i-1), \sigma(i+1), \ldots, \sigma(j), \sigma(i), \sigma(j+1), \ldots, \sigma(n)). \quad (10)$$

Further, the swap neighborhood comprises all sequences resulting from the interchange of two aircraft, so for $1 \leq i < j \leq n$ a swap neighbor of σ is

$$(\sigma(1), \ldots, \sigma(i-1), \sigma(j), \sigma(i+1), \ldots, \sigma(j-1), \sigma(i), \sigma(j+1), \ldots, \sigma(n)). \quad (11)$$

The 2-insert neighborhood comprises all sequences that can be obtained by removing two adjacent aircraft having the same weight class and inserting them into a new position in the sequence. Our motivation for this move type arises from the potential benefit of batching aircraft from the same weight class in terms of separation times. Note that these neighborhoods in combination can create solutions that cannot be formed by a merging of streams of pre-ordered aircraft as in our dynamic programing algorithm.

The descent algorithm uses the FCFS sequence as the initial solution, and selects a new solution using a *best improve* strategy when searching the combined insert, swap and 2-insert neighborhoods. Specifically, each iteration of the search generates all landing sequences that are neighbors of the current sequence, from which the corresponding (smallest) landing times are computed using (1) and (2). Any sequence that does not produce feasible landing times is not considered further, whereas other sequences with feasible landing times are evaluated using Eq. (7). The best neighbor is then selected. If it improves on the current solution, this best neighbor replaces the current solution and the search to improve the new current solution continues. If the best neighbor does not improve on the current solution, then the descent algorithm terminates with a local optimum.

Iterated descent prevents the descent algorithm from terminating at the first local optimum by applying a 'kick' to the locally optimal solution to create a new starting solution. Descent is then applied to this new solution, and the process repeats until a termination criterion ends the search process. Our kick corresponds to k randomly generated insert moves, where any such moves that cause infeasibility due to the latest landing time or maximum time shift constraints are rejected and consequently replaced by other random insert moves. We investigate different values of k in our computational experiments.

4.4. Simulated annealing

Simulated annealing is one of a number of local search techniques that can escape from local optima through accepting non-improving moves. In brief, the search randomly selects a neighbor, evaluates it with respect to the objective function, automatically accepts feasible improving neighbors and accepts feasible non-improving neighbors with a certain probability. A temperature parameter controls this probability, which dynamically changes through the search. The initial temperature is set so the probability of accepting non-improving moves is high, and as the search progresses the probability reduces. This is called the cooling schedule. Some researchers have investigated non-monotonic changes in temperature.

Our implementation of simulated annealing follows the approach proposed by Crauwels, Potts, and Wassenhove (1997) for scheduling families of jobs on a single machine, where a set-up time is required when the machine switches from processing a job in one family to a job in another family. There are some parallels with our problem, where aircraft are in families of classes and switching classes often incurs a greater separation than landing consecutive aircraft from the same class. We use the same three neighborhoods as in our iterated descent approach; insert, swap and 2-insert.

Neighbors producing a feasible solution with the same or a better objective function values than the current solution are accepted. On the other hand, feasible neighbors with a worse objective function value are accepted with probability $e^{-\Delta/t}$, where Δ is the amount by which the objective function increases and t is the temperature. We follow the scheme of Crauwels et al. (1997) in which the values of the temperature are periodic, rather than the usual scheme of starting with a high temperature which is gradually decreased during the course of the algorithm.

4.5. Dynamic problem

As explained in Section 3.4, the dynamic problem is based on solving a static problem every τ time units, where τ is the update time. The aircraft that are available to the static scheduling algorithm depend on two parameters in addition to τ. First, we consider the *time horizon* T over which the static problem is solved. Thus, at the update time, any aircraft that are within time T of the runway are assumed to be known to the ATC and are therefore included, but those aircraft with unconstrained landing times that are more than T time units into the future are excluded. Second, we assume that there is a *freeze time* t that defines the period of time for which the previously created schedule cannot be altered. As a consequence, any aircraft that are currently scheduled to land within the next t time units cannot be rescheduled. Note that the freeze time must exceed a certain minimum level to avoid potentially dangerous manoeuvres of aircraft that are close to the runway. Also, the time horizon T is selected to include all aircraft whose appearance times would reasonably be expected to be known to the ATC and might realistically influence scheduling decisions towards the start of the schedule that is to be generated.

An interesting observation is that the length of the time period $T - t$ is our main concern, rather than the specific values of t and T. For example, increasing both t and T by time τ will create the same schedule, but τ time units earlier than would be the case without the increase in t and T. Hence, we can set the length of the time window to be $T' = T - t$ with T' chosen such that knowledge of aircraft that are separated by more than $T - t$ time units does not significantly improve the quality of the landing schedule that is generated.

5. Computational experience

5.1. Heathrow test data

Our computational tests use two types of data sets. The first includes all landings at Heathrow Airport, UK, over a ten day period during June 2009. The second comprises data that are randomly generated in such a way to exhibit similar characteristics of traffic volume to the Heathrow data, and cover a 40-day period. The Heathrow data are the property of NATS (National Air Traffic Services) Ltd and subject to a non-disclosure agreement, hence motivating the generation of artificial data that can be made available to other researchers.

There are two parallel runways available for use at Heathrow airport. As the airport is situated close to residential areas, two runways generally operate in segregated mode; one for landing and one for take-off. Occasionally, landings are allowed on the nominated take-off runway to reduce delays and taxi times. Arriving aircraft approach from the east to west (westerly operation) unless the wind comes from the east in which case the landing direction is reversed so that aircraft land into the wind for safety reasons. During busy periods, controllers normally direct arriving aircraft to the top of one of four holding stacks. As aircraft reach the lowest level in their stack, controllers vector the aircraft onto the final approach and move higher aircraft down. Finally, they are merged into a single arrival stream of traffic for landing (website, 2011).

For the Heathrow data set, we extract information that is used to form the input into our scheduling algorithms and to provide a benchmark against which our algorithms are compared. Specifically, for each aircraft j, we find: the actual landing time, the landing runway, the aircraft's weight class (c_j) based on the UK's wake vortex group classification, date, the time that the aircraft crosses a cordon 40 nautical miles from the airport, and the unconstrained landing time (ult_j) as provided by Heathrow's arrival planner system. The UK has increased the original International Civil Aviation Organization's three wake turbulence separation groups to five, which in decreasing order of weight are Heavy (H), Upper medium (U), Lower medium (M), Small (S) and Light (L). The unconstrained landing time is calculated from the 40 nautical miles cordon crossing and provides the FCFS sequence that serves as an initial landing order. The crossing time of the 40 nautical miles cordon is the appearance time that defines when the flight becomes available to the controllers for scheduling. We use the landing runway data to identify and remove flights that do not land on the primary landing runway. Removing these data should not affect separation times for landing the other flights in the data set (although there may be implications on ATC workload but this is not considered in our model).

In addition to providing test instances for our algorithms, the Heathrow data are used to estimate the separation time matrix (s_{bc}) and to determine for each aircraft j its maximum time shift (ts_j). Note that ATCs are required to observe standard separation distances rather than times, and therefore the time between landings of aircraft is dependant on their approach speed.

In order to estimate separation times, we first extract the times between actual landings of consecutive flights. However, not all landings are queued, and consequently some separations may have greater than the minimum required. Hence, we remove any separation times that are greater than 1.2 times the standard separation distances divided by the estimated speed of aircraft immediately prior to landing at Heathrow. The remaining data are averaged by wake vortex leader/follower categories. Unfortunately, these data cannot be used directly because some categories have insufficient observations. Instead, we determine the airspeed that, when multiplied by the standard separation distances, gives the lowest mean square error from the separation times extracted from the Heathrow data set across all wake vortex categories. The separation times in Table 1 arise from a landing airspeed of 149 nautical miles per hour, which gives a mean squared error of 43.9.

Table 1
Separation times (seconds) based on an airspeed of 149 nautical miles per hour.

		Follower				
		H	U	M	S	L
Leader	H	97	121	121	145	169
	U	72	72	97	97	145
	M	72	72	72	72	121
	S	72	72	72	72	97
	L	72	72	72	72	72

Section 3 details the rationale for the maximum time shift, ts_j. Here, we define a common maximum time shift that applies to all aircraft. Analysis of the frequency of time shifts in the Heathrow data, after removing flights that land before their unconstrained landing time, show that 95% of the time shifts $\text{LT}_j - \text{ult}_j$ lie in the range 0–870 seconds after the unconstrained landing time. On this basis, we set $\text{ts}_j = 870$ seconds for all aircraft j. Since we do not have the necessary information to determine a meaningful preferred landing time, we assume that it is equal to the unconstrained landing time and therefore set $\text{plt}_j = \text{ult}_j$ for all aircraft j.

Finally, for our tests we set $\delta_j^e = 300$ and $\delta_j^l = 600$ for each aircraft j, which provides a 15-minute landing time window around preferred landing times during which no penalty is incurred. The values of the unit penalties u_j^e, u_j^l and v_j^l that appear in the objective function are not deducible from the Heathrow data; however, the values used in our computational tests are listed below in Section 5.3. Note that since $\text{LT}_j \geq \text{ult}_j = \text{plt}_j$ for each aircraft j, the value of δ_j^e (and u_j^e) is irrelevant.

5.2. Random test data

In order to generate the random test instances, we design a model that mimics the pattern of changes in traffic volume across the day and allows us to set different traffic intensities. As a result, we can evaluate the performance of the algorithms over a variety of problem instances. Each problem instance covers a one-day period. The appearance of the first aircraft is after 3 a.m. and the last aircraft before 10 p.m. Each day is divided into three periods: Morning (3–6 a.m), Day (6 a.m.–8 p.m.) and Night (8–10 p.m.). Fewer aircraft arrive during the Morning and Night periods. We further divide the Day period into Normal and Busy hours, where Busy hours are 6–8 a.m., 11 a.m.–1 p.m. and 4–7 p.m. and the remaining hours are Normal. Table 2 details the average number of aircraft μ per hour and the standard deviation σ, for each time period and each traffic intensity, where Set_1 represents the lowest intensity and Set_4 the highest. Ten instances are generated for each traffic intensity level, giving forty random test instances in total.

In addition to the number of flights, we also need a mechanism to generate for each aircraft j its weight class, appearance

Table 2
Means and standard deviations of hourly aircraft arrivals.

		Morning			Day		Night	
		3–4 a.m.	4–5 a.m.	5–6 a.m.	Normal	Busy	8–9 p.m.	9–10 p.m.
Set_1	μ	5	15	30	37	39	30	10
	σ	0.5	0.5	1.0	1.5	1.5	1.0	0.5
Set_2	μ	5	15	30	38	41	30	10
	σ	0.5	0.5	1.0	1.5	1.5	1.0	0.5
Set_3	μ	5	15	30	39	43	30	10
	σ	0.5	0.5	1.0	1.5	1.5	1.0	0.5
Set_4	μ	5	15	30	40	45	30	10
	σ	0.5	0.5	1.0	1.5	1.5	1.0	0.5

time (ap_j) and approach direction of the flight. For any time period t, probabilities $p_t(c)$ and $q_t(d)$ for the weight class $c \in$ {H, U, M, S, L} and approach direction $d \in \{1, \ldots, 10\}$ of an aircraft are derived from the 10-day Heathrow data, where d is the number of the dodecant corresponding to the position the aircraft crosses a cordon 40nm from the airport (only 10 of the 12 dodecants are used for approaches). Also, a negative exponential distribution provides a good fit for the inter-arrival time of flight appearance in the Heathrow data set.

The Daily Traffic Sample Generator below details the procedure for generating the test data. In brief, for each hour we generate the number of flights using the normal distribution $N(\mu, \sigma^2)$ based the means and standard deviations in Table 2. Then we generate the inter-arrival times between the flights using the negative exponential distribution. These times are scaled to ensure the arrivals exactly span the entire hour (with one aircraft appearing on the hour). The arrival times of the aircraft correspond directly to these values. The algorithm then computes further parameters for each aircraft j as follows. The weight class and approach direction are generated according to their respective probability distributions. Given d and the runway for landing, the remaining duration of the flight rdf_d, assuming an unimpeded passage to the runway, is estimated from the Heathrow data. Hence, we can calculate ult_j for each aircraft j. Finally, the latest landing time llt_j is found by randomly choosing a time gap of 1800, 2700 or 3600 seconds with probability 0.3, 0.5 and 0.2, respectively, and and adding it to the appearance time ap_j. Note that each of the time gaps exceeds the maximum time shift, and therefore latest landing times are effectively redundant when a maximum time shift constraint is imposed.

Daily Traffic Sample Generator

Execute the following steps for each hour $h = 1, \ldots, 19$ of the day, where the hours correspond to the time periods 3–4 a.m., …,9–10 p.m. Select the intensity Set_1, Set_2, Set_3 or Set_4 to be used, and set t to be one of the seven time periods according to the hour h and the columns of Table 2.

Generate appearance times

Generate the number of the aircraft n_h that appear during hour h from the normal distribution $N(\mu, \sigma^2)$, where μ and σ are given in Table 2.

Generate the gaps in seconds between aircraft appearances as follows.

A sample of unscaled inter-arrival times g_j in seconds for $j = 1, \ldots, n_h$ for hour h from an exponential distribution with mean $3600/n_h$ is generated.

Compute corresponding scaled inter-arrival times $\bar{g}_j = 3600 g_j / \sum_{i=1}^{n_h} g_i$ for $j = 1, \ldots, n_h$. Assign the appearance times in seconds using $ap_j = 3600(h-1) + \sum_{i=1}^{j} \bar{g}_i$ for $j = 1, \ldots, n_h$.

Generate data for each aircraft

Execute the following statements for each aircraft j, for $j = 1, \ldots, n_h$, that has an appearance time in hour h.

Generate a random number and use the the probabilities $p_t(c)$ for $c \in$ {H,U,M,S,L} to assign aircraft j a weight class.

Generate a random number and use the the probabilities $q_t(d)$ for $d \in$ {1,...,10} to assign aircraft j an approach direction (dodecant).

Generate a random number and set $l_j = ap_j + 1800$, $l_j = ap_j + 2700$ and $l_j = ap_j + 3600$ with probabilities 0.3, 0.5 and 0.2, respectively.

Set $ult_j = ap_j + rfd_d$, $plt_j = ult_j$, $\delta_j^e = 300$, $\delta_j^l = 600$, $ts_j = 870$, and use the entries in Table 3 below to assign values to u_j^e, u_j^l and v_j^l based on the weight class of aircraft j.

The generator does not guarantee that the resulting data set has a feasible schedule. Hence we check feasibility using dynamic programing and discard any set for which a feasible solution is not found. Typically this is because the time windows are too tight. The preferred landing times, maximum time shifts and values of the unit penalties used in the objective function are assigned values in the same way as for the Heathrow data, where the unit penalty values used in our computational tests are listed below in Section 5.3.

5.3. Experimental design

All algorithms were coded in MS Visual C++ 2008 and run on a PC with a dual core, 2.13 gigahertz and 2 gigabytes RAM. We refer to the first-come first-served, dynamic programing, iterated descent and simulated annealing algorithms as FCFS, Algorithm DP, Algorithm ID and Algorithm SA, respectively. For the static problem, our aim is to explore the performance of the different algorithms relative to FCFS and to landing schedules based on the landing sequence created by the ATCs. The solution of the static problem is typically of interest in strategic or tactical planning where the focus is to estimate the airport's capacity. However, for operational use, the main goal is to evaluate how well the dynamic problem is solved with the proposed algorithms.

For the static problem, we select three half-hour periods, three one-hour periods and one two-hour period to schedule aircraft from the 10-day Heathrow data set. These focus on time periods between 7–8 a.m., and 5–7 p.m., when demand for landing is particularly high. For the dynamic problem, each instance corresponds to the data for one day for both the Heathrow and random data sets, and each algorithm is run once for each instance. There are 10 instances for the Heathrow data. For the random data set, there are 10 instances for each of four sets of parameter values Set_1, Set_2, Set_3 and Set_4 as defined in Table 2, thus giving a total of 40 instances.

We now discuss some specific implementation details for Algorithms ID and SA when applied to the static problem. Both algorithms require an initial solution, which is provided by FCFS. An important parameter of Algorithm ID is the kick size. Based on results of initial experiments, we choose 6 randomly generated in-

Table 3
Weights for time window violation and extra fuel.

Weight class of j	H	U	M	S	L
u_j^l	20	17	15	12	10
v_j^l	15	13	12	10	8
u_j^e	10	8	7	5	4

sert moves (ignoring infeasible moves) as our kick. Algorithm ID and Algorithm SA terminate after 1, 2 and 4 seconds for $T = 30$, 60 and 120 minutes, respectively. Because both Algorithm ID and Algorithm SA employ randomization in some of their decisions, each algorithm is run $n/5$ times and the average performance is reported.

The implementation details for Algorithm ID and Algorithm SA for the dynamic problem are now outlined. The initial solution when creating an updated schedule is obtained by adding the newly available flights in FCFS order to the end of the previous schedule. As in the static problem, our initial experiments indicate that 6 randomly generated insert moves is an appropriate kick for Algorithm ID. The termination condition for each update in Algorithm ID and Algorithm SA is a computation time limit of 3 seconds because returning a solution quickly is critical. Algorithm ID and Algorithm SA are run once for each instance (the frequent schedule updates reduce the need for multiple runs).

For the dynamic problem, the previous schedule is updated every τ time units. We investigate the influence of τ by considering values $\tau = 2.5$, 5.0, 7.5 and 10 minutes. Scheduling starts after the freeze time that occupies the first t units of the scheduling period and considers those aircraft with unconstrained landing times that are no more than T time units into the future. As pointed out in Section 4.5, our interest is in the value of the parameter $T' = T - t$ that defines the length of the active time window. We investigate $T' = 10, 15, \ldots, 40$ minutes, and assume a freeze time of $t = 5$ minutes. For the Heathrow data, appearance time is between 13 and 20 minutes before unconstrained landing time depending on the approach route. In order to study longer active time windows, we subtract a constant $\max\{T' - 20, 0\}$ from the appearance times.

Our experiments investigate several sets of weight vectors (W_1, W_2, W_3, W_4) for the objective function defined in Eq. (7) of Section 3.2. When investigating throughput, we use the objective function defined in Eq. (4). For the full multi-criteria objective function (7), there are also penalties for time window violations and for the use of extra fuel if the unconstrained landing time is not achieved. Table 3 lists the unit penalty values that apply to each aircraft j within each of weight classes.

Our first weight vector is $W_1 = (0.3, 0.5, 0.1, 0.1)$, which reflects the throughput considerations of ATCs with some consideration of time-window violations and extra fuel cost. The second vector is $W_2 = (0.2, 0.4, 0.3, 0.1)$ which gives more emphasis to time-window violations. Note that delays relative to the time windows and extra fuel costs are non-conflicting. We also consider an objective that measures throughput, which is is given by the weight vector $W_3 = (0.4, 0.6, 0.0, 0.0)$ so that both $\mathrm{LT}_{\max}$ and ALT are considered. It is worth noting that ATCs typically prioritize throughput, and in particular $\mathrm{LT}_{\max}$, when deciding upon the landing order of aircraft. Hence, a throughput objective function is regarded as providing the best basis to compare our schedules against those designed by the controller. Although $\mathrm{LT}_{\max}$ provides the most natural measure of throughput, this objective only uses the landing time of the last aircraft. By ignoring other landing times $\mathrm{LT}_{\max}$ does not consider the first part of the schedule explicitly, which is the part that is actually used in the dynamic problem and not subject to update when new aircraft arrive into the system.

Our comparison of algorithms is based on the following performance statistics:

PI: percentage improvement in the value of the objective function provided by the algorithm relative to the initial (FCFS) sequence (off-line problem) or to a specific sequence (on-line problem);
TD: total deviation of the positions of aircraft in the landing sequence provided by the relevant algorithm from their positions in FCFS;
ND: number of aircraft with a change in position in the landing sequence provided by the relevant algorithm from the position in FCFS;
SEP: sum of the standard minimum separation times in seconds between aircraft implied by the solution sequence;
CT: computation time in seconds for scheduling a given set of aircraft for the off-line problem, or the computation time per update for the on-line problem.

As well as the overall weighted objective function, we also give values of PI relative to the individual components $\mathrm{LT}_{\max}$, ALT, TW and EF, where the latter three are defined in Eqs. (3), (5) and (6), respectively. The results for the Heathrow data also include those based on the ATC landing sequences, where earliest landing times are computed from ATC sequences by imposing earliest landing time constraints as given by (1) and the separation time constraints specified by the relevant inequality in (2). Using landing times computed in this way, rather than using actual landing times, allows a comparison using identical separation times to those used in our algorithms and is therefore fairer. TD and ND quantify deviations from the FCFS landing sequence, and therefore provide a measure of the amount of intervention necessary to achieve the landing schedule. SEP can be viewed as a measure of the amount of batching used to reduce separation times. Intuitively, reducing separation times is aligned with maximizing throughput; hence, ATCs implicitly use batching as a heuristic decision tool for increasing throughput.

5.4. Results

The tables that follow detail the average performance of the schedules arising from each of the approaches described in the earlier sections, where the objective function is defined by (7) for various choices of weights, and the time window constraint (1) and separation constraint (2) are imposed. As indicated above, results tables for the Heathrow data include ATC performance as measured by the landing times computed from the ATC's sequencing but using our separation times. As discussed previously, the ATC does not work to optimize our multi-objective function, and the data and constraints do not perfectly mirror the task that the ATC performs. Moreover, the minimum standard separations are currently based on distance (radar separation), which have been converted into time separations when used within our algorithms.

Tables 4 and 5 list average results for the Heathrow data used in a static environment, where each table corresponds to an alternative objective function. The first and second columns in each table give the durations in minutes of the time windows that define the aircraft to be scheduled and the average numbers of aircraft in the data set, respectively. The third column contains row headings for the objective function criteria. Hence, for each data set, the first row gives results for the main objective used by all of the approaches and the following four rows break down the objective function into its component criteria. For some entries in the TW row, a value of PI is not available (N/A) because the value of TW is zero for the initial FCFS sequence but positive for the algorithm under consideration.

Table 4, which is based on the objective function weight vector $W_1 = (0, 3, 0.5, 0.1, 0.1)$, shows an improvement in the main objective relative to the initial FCFS schedule across all of our

Table 4
Heathrow data, static environment: weights $W_1 = (0.3, 0.5, 0.1, 0.1)$ used in (7).

T	n	Obj.	ATC			Algorithm DP				Algorithm ID				Algorithm SA			
			PI	TD	ND	PI	TD	ND	CT	PI	TD	ND	CT	PI	TD	ND	CT
30	21	Overall	−0.72	15	10	1.47	15	8	0.16	1.47	13	7	1.00	1.47	15	8	1.00
		LT_{max}	0.02			0.07				0.07				0.07			
		ALT	0.01			0.07				0.07				0.07			
		TW	−5.42			2.65				2.65				2.65			
		EF	−3.46			8.36				8.36				8.36			
60	42	Overall	−0.84	36	22	4.41	35	16	0.56	4.41	29	15	2.00	4.41	31	16	2.00
		LT_{max}	0.00			0.19				0.19				0.19			
		ALT	−0.15			0.12				0.12				0.12			
		TW	−7.16			16.16				16.16				16.16			
		EF	−4.07			10.50				10.50				10.50			
120	84	Overall	−0.81	72	47	7.08	74	30	7.73	7.05	78	34	4.00	7.05	84	36	4.00
		LT_{max}	0.00			0.00				0.00				0.00			
		ALT	0.00			0.09				0.08				0.08			
		TW	N/A			N/A				0.00				0.00			
		EF	−8.12			24.27				23.65				23.65			

Table 5
Heathrow data, static environment: weights $W_3 = (0.4, 0.6, 0.0, 0.0)$ used in (7).

T	n	Obj.	ATC			Algorithm DP				Algorithm ID				Algorithm SA			
			PI	TD	ND	PI	TD	ND	CT	PI	TD	ND	CT	PI	TD	ND	CT
30	21	Overall	0.01	15	10	0.10	22	9	0.25	0.10	18	7	1.00	0.10	14	6	1.00
		LT_{max}	0.02			0.13				0.13				0.13			
		ALT	0.01			0.08				0.08				0.08			
60	42	Overall	0.09	36	22	0.18	63	23	0.72	0.18	43	18	2.00	0.18	61	22	2.00
		LT_{max}	0.00			0.23				0.23				0.23			
		ALT	0.15			0.14				0.14				0.14			
120	84	Overall	0.00	72	47	0.05	88	34	7.64	0.05	56	22	4.00	0.05	96	34	4.00
		LT_{max}	0.00			0.00				0.00				0.00			
		ALT	0.00			0.09				0.08				0.09			

algorithms. The ATC schedule does not show an improvement, but this is largely due to the TW and EF cost elements. This is expected since there is no attempt by the ATC to reduce lateness or the cost of fuel. However, there is some degradation in ALT for $T = 60$, which is explored in more detail below in the discussion for dynamic environment. When considering the breakdown of criteria, the improvement for LT_{max} and ALT is modest; it is clear that our approaches are producing similar throughput but with reduced cost associated with time window violations and extra fuel. Results for the objective function weight vector $W_2 = (0, 2, 0.4, 0.3, 0.1)$ are not displayed because they exhibit a similar pattern to those in Table 4. Table 5 displays results for an objective function with weighted vector $W_3 = (0.4, 0.6, 0, 0)$ that does not consider costs for time window violations or extra fuel. The values in the PI columns show that small average improvements are typically observed over FCFS.

Comparing across the different solution approaches, dynamic programing has the most variable computation times and these times become longer for the larger instances with $T = 120$. For the data instances considered, greater computational effort does not lead to improved schedules, with Algorithms DP, ID and SA having similar averages for the objective functions in Tables 4 and 5. The ability of Algorithms ID and SA to obtain competitive solutions quickly indicates that they are well suited for use in a dynamic environment.

One statistic of note is that all of our approaches are finding schedules of similar performance across all data instances, but with varying deviations from the initial FCFS sequence as measured by TD and ND. This suggests that there are multiple local optima with similar objective function values.

We now present our computational results for the dynamic environment. For these tests, we again compare Algorithms DP, ID and SA since their performance in terms of the quality of schedules generated for the static problem is similar. Note that the reported results are based on schedules created for a complete day.

Table 6
Influence of τ: average PI relative to $\tau = 2.5$ minutes.

Dataset	Update time τ (minutes)		
	5	7.5	10
10-day Heathrow	0.011	−0.029	−0.050
8-day random data	0.000	−0.052	−0.297

Table 7
Influence of T': average PI relative to $T' = 15$ minutes.

Dataset	Measure	Active scheduling window length T' (minutes)					
		15	20	25	30	35	40
10-day Heathrow	Ave. PI	0.000	0.743	0.931	0.999	1.022	0.983
	Ave. CT	0.002	0.020	0.103	0.308	0.883	1.959
8-day random data	Ave. PI	0.000	0.245	0.761	0.855	0.973	0.980
	Ave. CT	0.002	0.012	0.052	0.157	0.380	0.831

We design an initial set of experiments in order to test parameters using the Heathrow data, and eight days of the random test data, where two days are randomly chosen for each of the four traffic intensity levels. We first experiment with the update times $\tau = 2.5$, 5.0, 7.5 and 10.0 minutes, where dynamic programing is applied to solve the resulting problem at each update and the objective function is defined by the weight vector (0.3, 0.5, 0.1, 0.1). Table 6 lists average PI values that are evaluated relative to objective function values obtained with $\tau = 2.5$ minutes, $t = 5$ minutes and $T = 30$ minutes. It is clear from the results that $\tau = 5$ minutes provides the best strategy in terms of solution quality, and it

Table 8
Heathrow data, dynamic environment: Average PI relative to FCFS.

Weight	Obj	FCFS	ATC				Algorithm DP						Algorithm ID				Algorithm SA			
		Sep.	PI	TD	ND	Sep	PI	TD	ND	CT	Max-CT	Sep	PI	TD	ND	Sep	PI	TD	ND	Sep
W_1	Overall	54151	−5.55	569	302	53354	23.76	504	218	0.11	58.60	52838	23.60	523	222	52840	22.60	556	225	53000
	ALT		0.00				0.14						0.14				0.13			
	TW		−463.65				34.24						32.70				55.52			
	EF		−2.19				29.46						29.33				27.92			
W_2	Overall	54151	−5.33	569	302	53354	26.32	486	216	0.10	52.90	52869	26.19	511	220	52871	25.22	534	219	52960
	ALT		0.00				0.13						0.13				0.13			
	TW		−463.65				72.17						70.04				77.47			
	EF		−2.19				46.93						28.59				27.01			
W_3	Overall	54151	0.00	569	302	53354	0.05	736	272	0.09	26.40	52944	0.07	721	266	52906	0.07	742	268	53002
	ALT		0.00				0.11						0.15				0.14			
	TW		−463.65				−538.30						−328.24				−377.82			
	EF		7.79				18.14						29.37				27.18			

Table 9
Random data, dynamic environment: average PI relative to FCFS.

Weight	Obj	Algorithm DP					Algorithm ID			Algorithm SA		
		PI	TD	ND	CT	Max-CT	PI	TD	ND	PI	TD	ND
W_1	Overall	26.53	527	266	0.11	68.70	26.50	572	277	25.48	556	265
	ALT	0.20					0.20			0.19		
	TW	59.69					60.09			72.39		
	EF	29.09					29.04			27.66		
W_2	Overall	31.32	503	261	0.14	77.50	31.26	544	271	30.13	533	260
	ALT	0.19					0.19			0.18		
	TW	78.81					78.66			80.27		
	EF	28.33					28.28			26.88		
W_3	Overall	0.10	754	320	0.11	71.60	0.10	798	326	0.09	807	323
	ALT	0.21					0.21			0.20		
	TW	−573.23					−747.66			−737.47		
	EF	29.24					29.13			27.35		

has a lower computational requirement than the next best value of $\tau = 2.5$ minutes.

Having decided on the value $\tau = 5$ minutes, we now compare various active time window lengths $T' = 15,\ 20,\ 25,\ 30,\ 35$ and 40 minutes. Again, we use an objective function defined with the weight vector $W_1 = (0.3, 0.5, 0.1, 0.1)$ and apply dynamic programming at each update. Table 7 lists average PI values that are evaluated relative to objective function values obtained with $T' = 15$ minutes. These results show that the quality of schedules and the computation time increase as T' becomes larger. However, the improvement in solution quality becomes less with each five minute lengthening of T', whereas the computation time significantly increases. Hence, there are rapidly diminishing returns after $T' = 25$. The results also confirm our intuition that new aircraft introduced, which are not close enough to the runway to land until later in the schedule, are unlikely to affect the order of the aircraft towards the beginning of the schedule especially for larger values of T'. As a result, we select $T' = 25$ minutes as the length of active time window.

Tables 8 and 9 detail the full results for the dynamic problem with $\tau = 5,\ T' = 25$. The tables compare the schedules obtained from the ATC sequence (only for the Heathrow data in Table 8) and the schedules obtained with updates using Algorithm DP, Algorithm ID with $k = 6$, and Algorithm SA, where the latter two algorithms have a computation time limit of 3 seconds for each update. For Algorithm DP, the average time for updating the schedule is provided in the column CT, and the maximum time in the column Max-CT

Note that we do not include LT_{max} when reporting on the PI values for the individual objective functions components because the landing time of the days last aircraft is not a meaningful performance measure.

The results for the Heathrow data in Table 8 show that dynamic programing, iterated descent and simulated annealing provide schedules that improve over FCFS and ATC for all objective functions and all of its components except for TW when the objective function weights are vector is $W_3 = (0.4, 0.6, 0, 0)$. As with the static case, the schedules obtained from the ATC sequence appear to be inferior to FCFS schedules. However, TW and EF play a major role in the reduction in solution quality; neither of them are used by ATC in making scheduling decisions. Our understanding is that ATC seek to maximize runway utilization by reducing separation times. This can be achieved locally by batching aircraft. Performance measure Sep sums the minimum separation time between consecutive aircraft given the landing sequence. Using this measure, it is clear that ATC are successfully reducing separation times relative to FCFS. Algorithms DP, ID and SA also improve on FCFS and ATC by this measure.

Table 9 shows a similar performance using the randomly generated test data. For both the Heathrow and the random data, Algorithms DP, ID and SA exhibit similar levels of performance. However, DP has variable computation times and could be vulnerable if many new aircraft appear in the same update period. SA is harder to implement than ID due to the many parameters that need to be tuned, while ID only requires the parameter k to be specified. Overall, Algorithm ID has slight advantages over the two other competing methods.

6. Concluding remarks

This paper has introduced models and algorithms for the static/off-line aircraft landing problem and the dynamic/on-line version of the problem. A special feature of our model is the multi-objective approach that takes into account the agendas of the

various stakeholders that have an interest in the scheduling of landing aircraft.

Dynamic programing, iterated descent and simulated annealing algorithms are proposed for the static problem. Also, using a rolling horizon approach, the dynamic problem is tackled by periodically updating the previous schedule with an iterated descent or dynamic programing solution approach. A thorough computational evaluation is performed using data from Heathrow airport and randomly generated test data.

Results for the static problem show that all of the proposed algorithms are effective in achieving an efficient runway throughput. In addition, the algorithms are capable of finding solutions that perform well in terms of minimizing delay and minimizing the cost of extra fuel used to achieve the desired landing schedule. Iterated descent has the advantage of being faster and having more predictable run times than the other approaches, and is therefore preferred to dynamic programing and simulated annealing.

For the dynamic problem, the frequency of update time and the length of the time window when aircraft are available for scheduling are investigated. A five minute update time provides as good solutions as with a more frequent update, and has a lower computational cost. A time window of twenty-five minutes for scheduling is chosen. Wider time windows have diminishing returns and require much greater computational effort. Our overall computational results show that iterated descent and dynamic programing provide schedules that improve upon FCFS across all objective function elements. However, iterated descent is preferred to dynamic programing because of its more modest and predictable computational requirements.

Acknowledgments

The authors are grateful to Alan Drew and Paul Humphreys of EUROCONTROL for their help and support and to John Greenwood of NATS for providing the Heathrow date that formed the basis for our computational evaluation of algorithms. We greatly appreciate the useful advice from anonymous referees that resulted in the exposition of the paper being improved.

This work has been co-financed by the European Organization for the Safety of Air Navigation (EUROCONTROL) under its Research Grant scheme (CARE Grants 0 8-120920-C).

The content of the work does not necessarily reflect the official position of EUROCONTROL on the matter.

References

Abela, J., Abramson, D., Krishnamoorthy, M., & Silva, A. D. (1993). Computing optimal schedules for landing aircraft. *Proceedings of the twelfth national conference of the australian society for operations research* (pp. 71–90). Australia: Adelaide. July 7–9

Artiouchine, K., Baptiste, P., & Dürr, C. (2008). Runway sequencing with holding patterns. *European Journal of Operational Research, 189*, 1254–1266.

Balakrishnan, H., & Chandran, B. (2010). Scheduling aircraft landings under constrained position shifting. *Operations Research, 58*, 1650–1665.

Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., & Abramson, D. (2000). Scheduling aircraft landings – the static case. *Transportation Science, 34*, 180–197.

Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., & Abramson, D. (2004). Displacement problem and dynamically scheduling aircraft landings. *Journal of the Operational Research Society, 55*, 54–64.

Beasley, J. E., Sonander, J., & Havelock, P. (2001). Scheduling aircraft landing at london heathrow using a population heuristic. *Journal of the Operational Research Society, 52*, 483–493.

Bennell, J. A., Mesgarpour, M., & Potts, C. N. (2011). Airport runway scheduling. *Annals of Operations Research, 204*, 249–270.

Brentnall, A. R. (2006). *Aircraft arrival management*. UK: University of Southampton Ph.D. thesis.

Chandran, B., & Balakrishnan, H. (2007). A dynamic programming algorithm for robust runway scheduling. In *Proceedings of the american control conference, New York, USA* (pp. 1161–1166).

Ciesielski, V., & Scerri, P. (1997). An anytime algorithm for scheduling of aircraft landing times using genetic algorithms. *Australian Journal of Intelligent Information Processing Systems, 4*, 206–213.

Crauwels, H. A. J., Potts, C. N., & Wassenhove, L. N. V. (1997). Local search heuristics for single machine scheduling with batch set-up times to minimize total weighted completion time. *Annals of Operations Research, 70*, 261–279.

D'Ariano, A., Pacciarelli, D., Pistelli, M., & Pranzo, M. (2015). Real-time scheduling of aircraft arrivals and departures in a terminal maneuvering area. *Networks, 65*, 212–227.

D'Ariano, A., Pistelli, M., & Pacciarelli, D. (2012). Aircraft retiming and rerouting in vicinity of airports. *IET Intelligent Transport Systems, 6*, 433–443.

Dear, R. (1976). The dynamic scheduling of aircraft in the near terminal area. *Technicla report, r76-9*. MIT, USA: Flight Transportation Laboratory.

Ernst, A. T., Krishnamoorthy, M., & Storer, R. H. (1999). Heuristic and exact algorithms for scheduling aircraft landings. *Networks, 34*, 229–241.

Fahle, T., Feldmann, R., Götz, S., Grothklags, S., & Monien, B. (2003). The aircraft sequencing problem. In R. Klein, H. W. Six, & L. Wegner (Eds.), *Computer science in perspective, LNCS 2598* (pp. 152–166). Berlin: Springer-Verlag.

Faye, A. (2015). Solving the aircraft landing problem with time discretization approach. *European Journal of Operational Research, 242*, 1028–1038.

Hu, X. B., & Chen, W. H. (2005a). Receding horizon control for aircraft arrival sequencing and scheduling. *IEEE Transaction on Intelligent Transportation Systems, 6*, 189–197.

Hu, X. B., & Chen, W. H. (2005b). Genetic algorithm based on receding horizon control for arrival sequencing and scheduling. *Engineering Applications of Artificial Intelligence, 18*, 633–642.

Lee, H., & Balakrishnan, H. (2008). Fuel cost, delay and throughput tradeoffs in runway scheduling. *Proceeding of american control conference (ACC 08)* (pp. 1161–1166). Washington, USA: Seattle.

Moser, I., & Hendtlass, T. (2007). Solving dynamic single-runway aircraft landing problems with extremal optimization. *Proceeding of the 2007 IEEE symposium on computational intelligence in scheduling (CI-Sched 2007)* (pp. 206–211). Hawaii, USA: Honolulu.

Murça, M. C. R., & Müller, C. (2015). Control-based optimization approach for aircraft scheduling in a terminal area with alternative arrival routes. *Transportation Research Part E, 73*, 96–113.

Pinol, H., & Beasley, J. E. (2006). Scatter search and bionomic algorithms for the aircraft landing problem. *European Journal of Operational Research, 171*, 439–462.

Psaraftis, H. N. (1978). A dynamic programming approach to the aircraft sequencing problem. *Technical report, r78-4*. MIT, USA: Flight Transportation Laboratory.

Psaraftis, H. N. (1980). A dynamic programming approach for sequencing groups of identical jobs. *Operations Research, 28*, 1347–1359.

Samà, M., D'Ariano, A., D'Ariano, P., & Pacciarelli, D. (2014). Optimal aircraft scheduling and routing at a terminal control area during disturbances. *Transportation Research Part C, 47*, 61–85.

Samà, M., D'Ariano, A., D'Ariano, P., & Pacciarelli, D. (2016). Scheduling models for optimal aircraft traffic control at busy airports: Tardiness, priorities, equity and violations considerations. *Omega*. doi:10.1016/j.omega.2016.04.003.

Samà, M., D'Ariano, A., & Pacciarelli, D. (2013). Rolling horizon approach for aircraft scheduling in the terminal control area of busy airports. *Transportation Research Part E, 60*, 140–155.

Soomer, M. J., & Franx, G. J. (2008). Scheduling aircraft landings using airlines' preferences. *European Journal of Operational Research, 190*, 277–291.

website, H. A. o. (2011). Retreived sept 1st. Available at: http://www.heathrowairport.com.